Opt-AI.

Weekly Seminar

Opt-AI. LLM Research Team

Exaone 2.4B ONNX → QNN Hands on example

Table of Contents

01
Review Last Lessons

Quiz

1. LLM에서 “RLHF”는 무엇을 의미하며, 왜 중요한가요?

---

1. Reinforcement Learning with High Fidelity
2. Reinforcement Learning with Human Feedback
3. Recurrent Learning with Human Feedback
4. Reward Learning with Human Feedback

Quiz

2. Multimodal Model이란 무엇인가요?

---

1. 단일 데이터 유형만 학습하는 모델
2. 여러 유형의 입력 데이터를 동시에 다루는 모델
3. Multi GPU에서만 동작하는 모델
4. 정형 데이터에 특화된 모델

Quiz

3. LoRA 기법의 주요 이점은 무엇인가요?

---

1. 고정된 학습률을 사용하는 것으로 메모리를 절약한다
2. Attention 메커니점을 제거하여 간소화된 학습 구조를 제공한다
3. 기존 모델의 가중치를 Low-rank로 근사하여 효율적인 Fine-tuning 제공
4. 모든 파라미터를 동결하여 더 빠르게 Fine-tuning할 수 있다

Quiz

4. LLM에서 Quantization의 주요 목적은 무엇인가요?

---

1. 학습 데이터를 변환하기 위해
2. 모델의 품질을 향상시키기 위해
3. 모델 크기를 줄이고 추론 속도를 높이기 위해
4. 다양한 데이터를 학습하기 위해

Quiz

5. CLIP의 주요 기능은 무엇인가요?

---

1. 이미지를 처리하는 모델
2. 텍스트와 이미지 간의 연관성을 학습하는 모델
3. 단일 언어 데이터를 학습하는 모델
4. 구조적 데이터를 분석하는 모델

Quiz

6. Transformer 기반 LLM에서 Multi-head Attention의 주요 역할은 무엇인가요?

1. 입력 데이터의 순서를 무시
2. 다양한 입력 간의 관계를 병렬로 분석
3. 메모리 사용량을 줄이기 위해 그래디언트를 여러 개로 나눔
4. 모델의 크기를 줄이기 위한 파라미터 감소

Quiz

7. Fixed Point와 Floating Point의 주요 차이점은 무엇인가요?

---

1. Floating point는 정수 데이터를 처리할 수 없다
2. Fixed point는 소수점 위치가 고정되어 있고, Floating point는 가변적이다
3. Fixed point는 하드웨어 비효율적이다
4. Floating point는 Uniform distribution에 적합함

Quiz

8. Language Model의 주요 분류 중 BERT 계열 모델은 무엇에 주로 사용되나요?

1. 텍스트 생성
2. 문장 분류 및 응답
3. 이미지 분류
4. 기계 번역

Quiz

9. Tokenizer에서 BPE(Byte Pair Encoding) 방식의 주요 목적은 무엇인가요?

1. 모든 단어를 개별 토큰으로 변환
2. 단어를 문자 단위로 분리
3. 자주 등장하는 문자 쌍을 병합해 효율적인 토큰화
4. 희귀 단어를 제거하여 데이터 크기를 줄임

Quiz

10. LLM에서 Context Length를 증가시킬 때 발생할 수 있는 주요 문제는 무엇인가요?

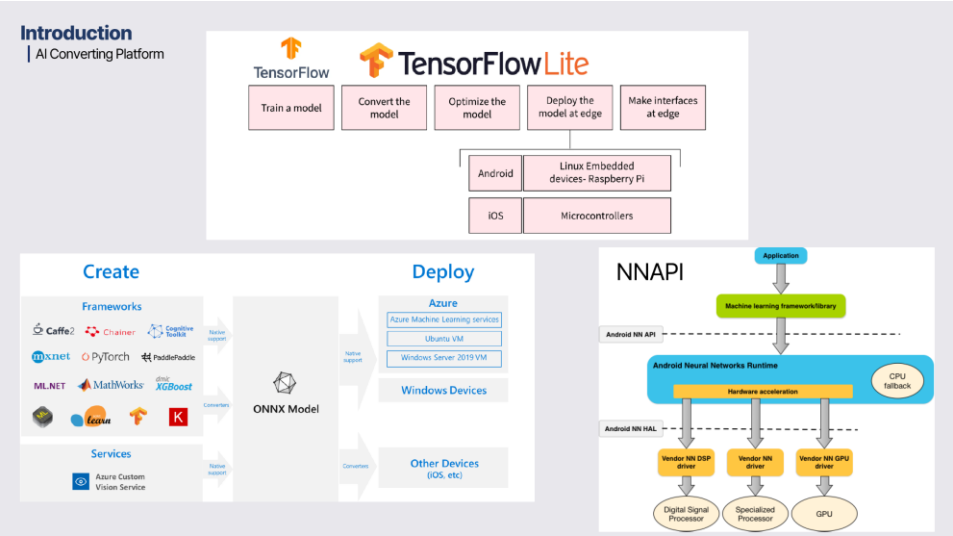
---

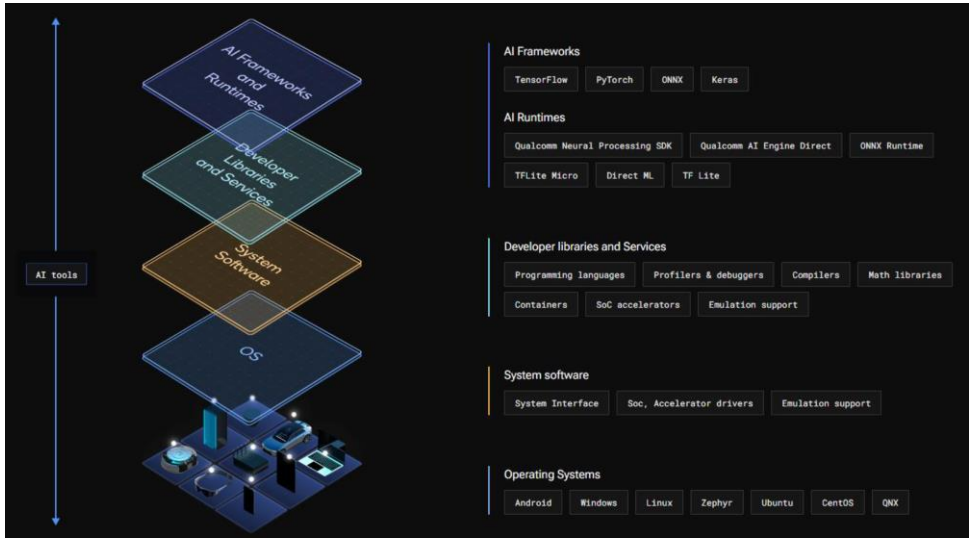
1. 긴 문맥을 더 잘 이해할 수 있다
2. 모델의 추론 속도가 빨라진다
3. KV Cache 크기가 커지면서 메모리 사용량이 증가한다
4. 긴 Context Length로 인해 데이터 전처리가 단순해진다

Review

Week 4 ~ 7

Week 4



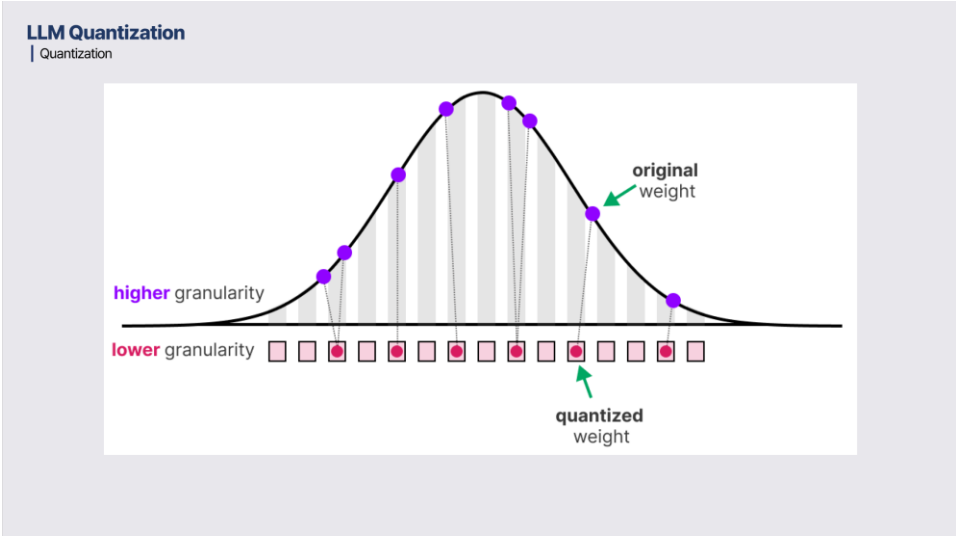


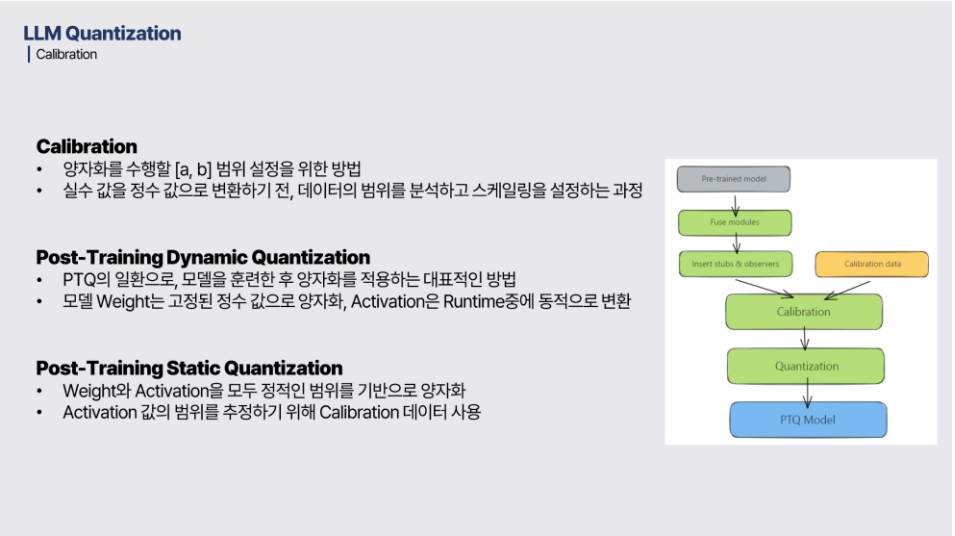
- On-Device AI Platform의 종류: TFLite, ONNX, NNAPI, Executorch, ...
- 제조사 별 SoC의 차이점: Snapdragon, Exynos, ...
- Executorch export process에 대한 코드 리뷰

Review

Week 4 ~ 7

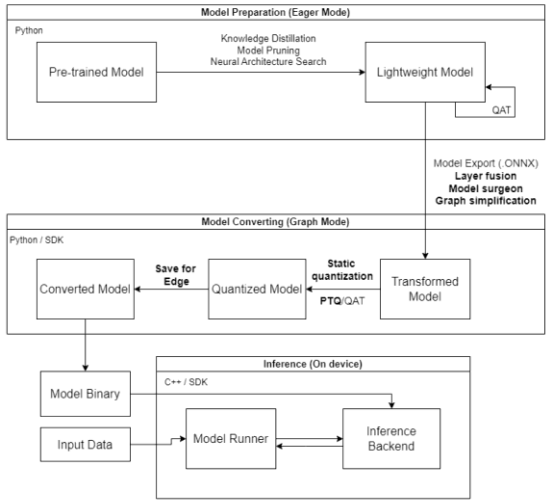
Week 5

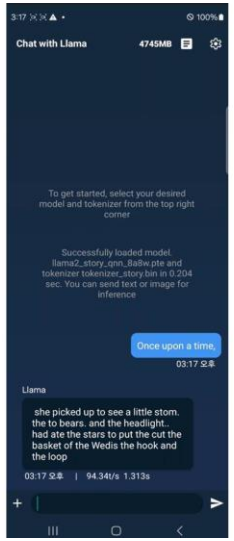


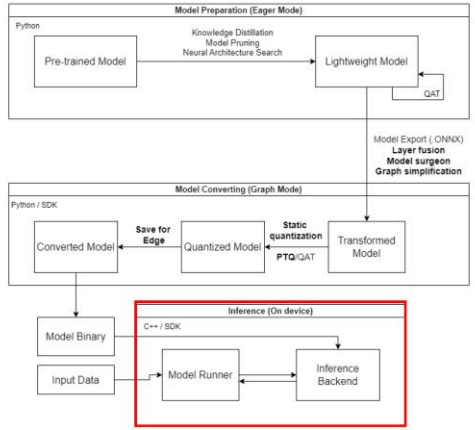


- Static Quantization과 Dynamic Quantization의 차이점
- 다양한 양자화 기법에 대해 학습
- Quantization에 Calibration의 필요성
- Executorch 기반 QNN Backend Export 실습

Week 6







- On-Device Model 변환을 위한 과정 Review
- Llama-3.2-1B-Instruct 모델을 XNNPACK, QNN, Quantization 등 다양한 방법으로 변환 시도
- QNN Quantization을 수행하였을 때, Calibration 유무의 차이 확인

Review

Week 4 ~ 7

Week 7

MultiModal

What is Multimodal

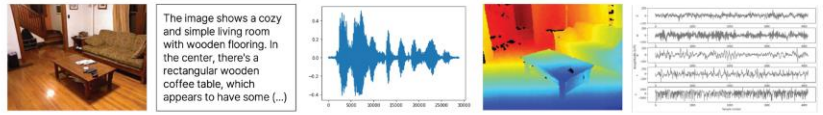
What is Multimodal

Modal

- 한국어로는 양식, 양상이라 하며, 어떤 일을 하거나 경험하는 특정한 방식
- 시각, 촉각, 청각 등 인간의 다양한 감각을 의미함
- 인공지능에서는 입력으로 사용되는 모든 데이터의 양상을 일컬음

Multimodal

- 하나의 모델이 두 개 이상의 서로 다른 모달리티를 다루는 방식



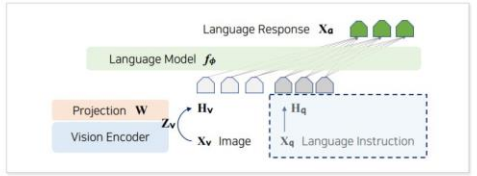
이미지 텍스트 오디오 깊이 신호

Llava

What is Llava

LLaVA

- LLaVA: Pre-trained LLM + Pre-trained Visual Model
  - ✓ LLM: Vicuna, Llama
  - ✓ Visual Model: CLIP



< LLaVA Architecture >

- LLaMA를 비롯한 LLM의 전체적인 내용 Review
- Multimodal learning 개념 및 필요성
- Apple Intelligence를 통한 MLM의 필요성
- LLaVA 실습

02
Is Executorch Enough?

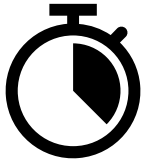
Executorch?

Reasons to avoid

- Environmental Issue
  - Limited target SoC
  - Hard to control deployment

Limited model support

- Supports only Llama
- Supports only torch-based model



Restricted Optimization

- No Graph Editing
- No Layer fusion without torch code

Stability

- Rapid change
- Possibility of discontinuing

Executorch?

Another Options

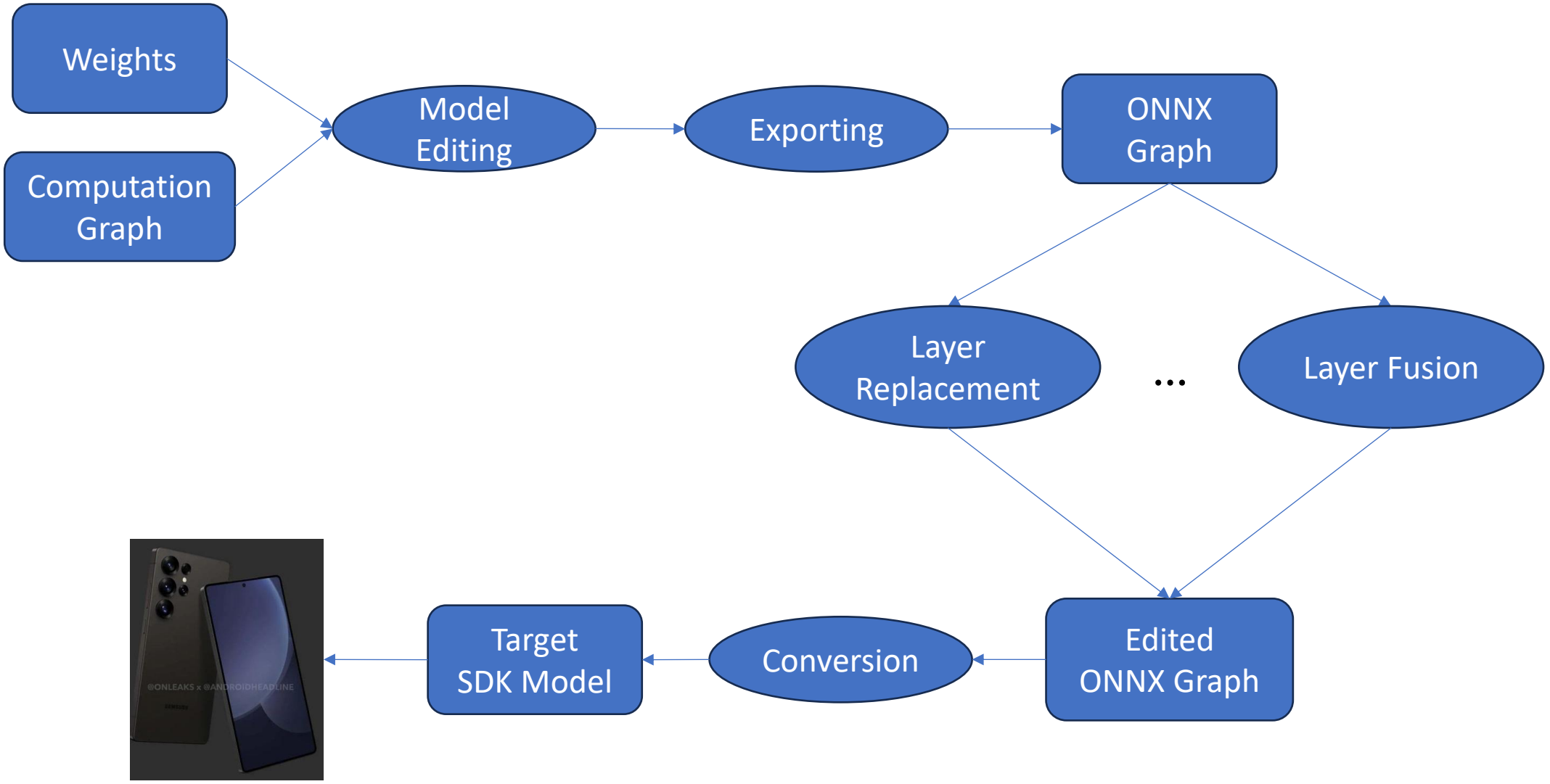
When Executorch isn't enough, which alternative is best?

Options	Key Features	pros	cons	Support Platform	Relative Speed	Accessibility	Reference
ONNX-runtime	Framework interoperability	Fast Deployment	Not Optimized	Independent	Slow	MIT	ONNX-runtime
NCNN	Lightweight, mobile-friendly	Cross platform	High man month	CPU / Vulkan	Average	BSD-3	NCNN
QNN	Optimized for Qualcomm hardware	Optimized for Target SoC	Limited to Qualcomm SoC	Qualcomm SoC	Fast	Private (승인 필요)	QNN
TensorRT	High-performance GPU optimization	Optimized for Target SoC	Limited to Nvidia SoC	Nvidia Soc	Fast	Private	TensorRT

03
QNN with ONNX Framework

QNN With ONNX

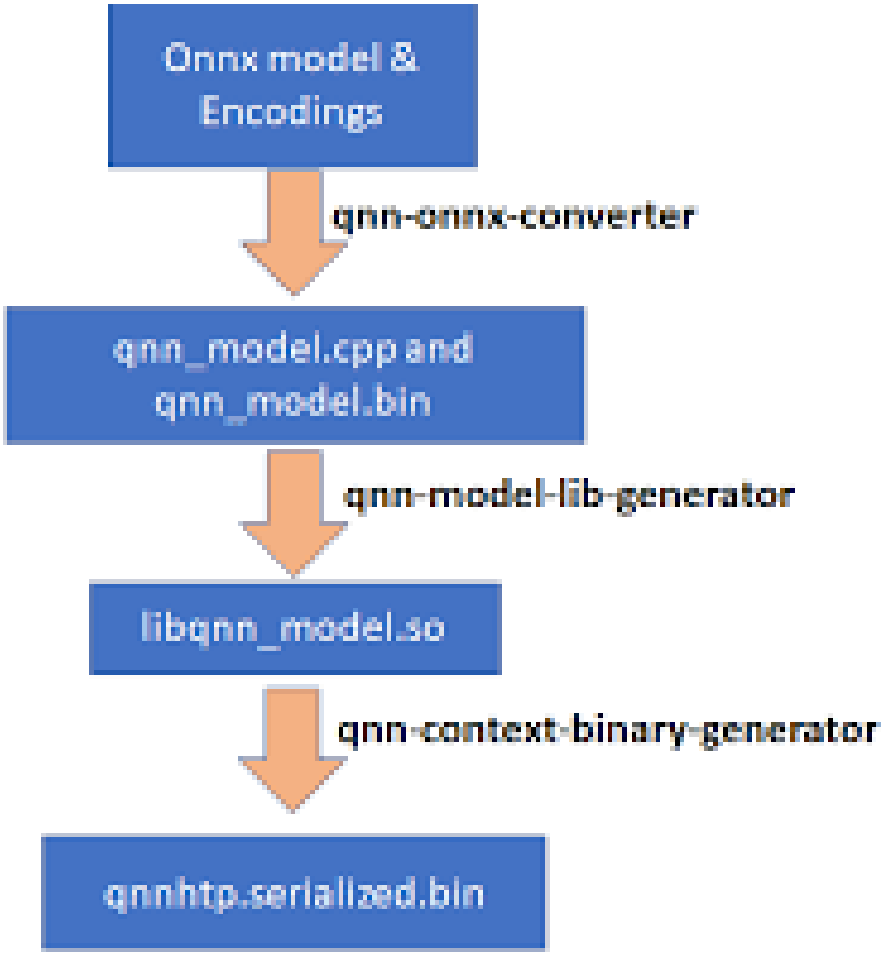
Preliminaries



QNN With ONNX

Preliminaries

```
QNN_API
ModelError_t QnnModel_composeGraphs(...){
  ModelError_t err = MODEL_NO_ERROR;

  /* model/graph for norm*/
  QnnModel norm;
  const QnnGraph_Config_t** graphConfigs = nullptr;
  ....
  VALIDATE(addNode_node_Mul_7(norm), err);

  // Add all models to array to get graphsInfo
  QnnModel* models [] = {&norm};
  uint32_t numModels = 1;

  // Populate the constructed graphs in provided output variables
  VALIDATE(getGraphInfoFromModels(*models, numModels, graphsInfo), err);
  *numGraphsInfo = numModels;

  return err;

} // PREPARE_GRAPHS
```



QNN With ONNX

Model Editing

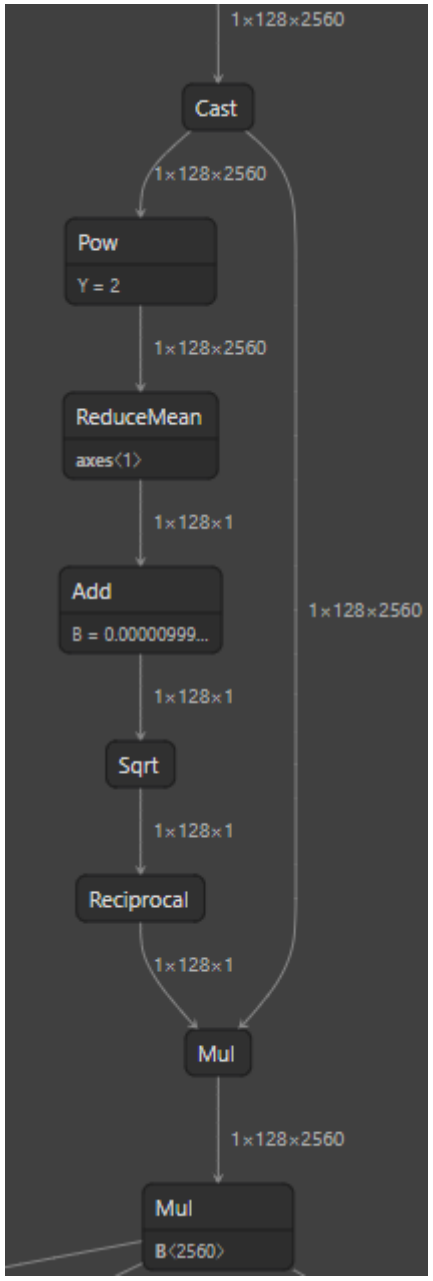
```
checkpointer = FullModelHFCheckpointer(
    checkpoint_dir = model_path,
    checkpoint_files = [
          'model-00001-of-00002.safetensors',
        'model-00002-of-00002.safetensors'
        ],
    output_dir = model_path + '/output' ,
    model_type = 'LLAMA2'
)
sd = checkpointer.load_checkpoint()
model = torchtune.models.llama2.llama2(64000, 32, 32, 8, 2560, 128, 0, 7168)

model.load_state_dict(sd['model'])
```

$$\text{rms}(\mathbf{x}) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}$$

$$\hat{x}_i = \frac{x_i}{\text{rms}(\mathbf{x}) + \epsilon}$$

$$y_i = \gamma \hat{x}_i$$

Why?
Avoidable?



With original model, fp16 exporting will fail

0.0ms [ERROR] tcm_migration.cc:1863:ERROR:no properties registered for q::QNN_RmsNorm
@qnn-context-bin-generator

QNN With ONNX

Model Editing

If then Replace it! But.. Are we done?

```python
class Normalization(nn.Module):
  def __init__(self, shape, scale: nn.Parameter, *args, **kwargs) -> None:
    super().__init__(*args, **kwargs)
    self.norm = nn.LayerNorm(shape, elementwise_affine=False, bias=False)
    #self.norm = nn.GroupNorm(1, scale.shape[0], affine=False)
    self.scale = scale

  def forward(self, x: torch.Tensor):
    return self.norm(x) * self.scale

def replace_rmsnorm_with_layernorm(model):
    for name, module in model.named_children():
        if isinstance(module, RMSNorm):  # Assuming RMSNorm is defined somewhere
            # Create a new LayerNorm with the same number of features as the RMSNorm layer
            setattr(model, name, Normalization([*module.scale.shape], module.scale))
        else:
            # Recursively replace RMSNorm in submodules
            replace_rmsnorm_with_layernorm(module)
```

구분	Layer Normalization	RMSNorm
정규화 방식	평균과 분산을 사용하여 정규화	제곱평균을 사용하여 정규화
계산 비용	평균과 분산 계산이 필요하여 비용이 더 높음	제곱평균 계산만 필요하여 비용이 더 낮음
파라미터	γ (스케일 파라미터), β (시프트 파라미터)	γ (스케일 파라미터)
배치 민감도	민감하지 않음	민감하지 않음
주 사용 사례	RNN과 같은 순환 신경망	대규모 신경망, 계산 비용이 낮은 모델 (transformer)
장점	각 샘플의 평균과 분산을 고려하여 정규화하여 더 안정적인 학습 가능	계산 비용이 낮고 간단한 계산으로 정규화 수행 가능
단점	평균과 분산 계산으로 인해 계산 비용이 높음	평균을 고려하지 않기 때문에 일부 경우에서 학습 안정성 떨어질 수 있음

QNN With ONNX

Model Partitioning

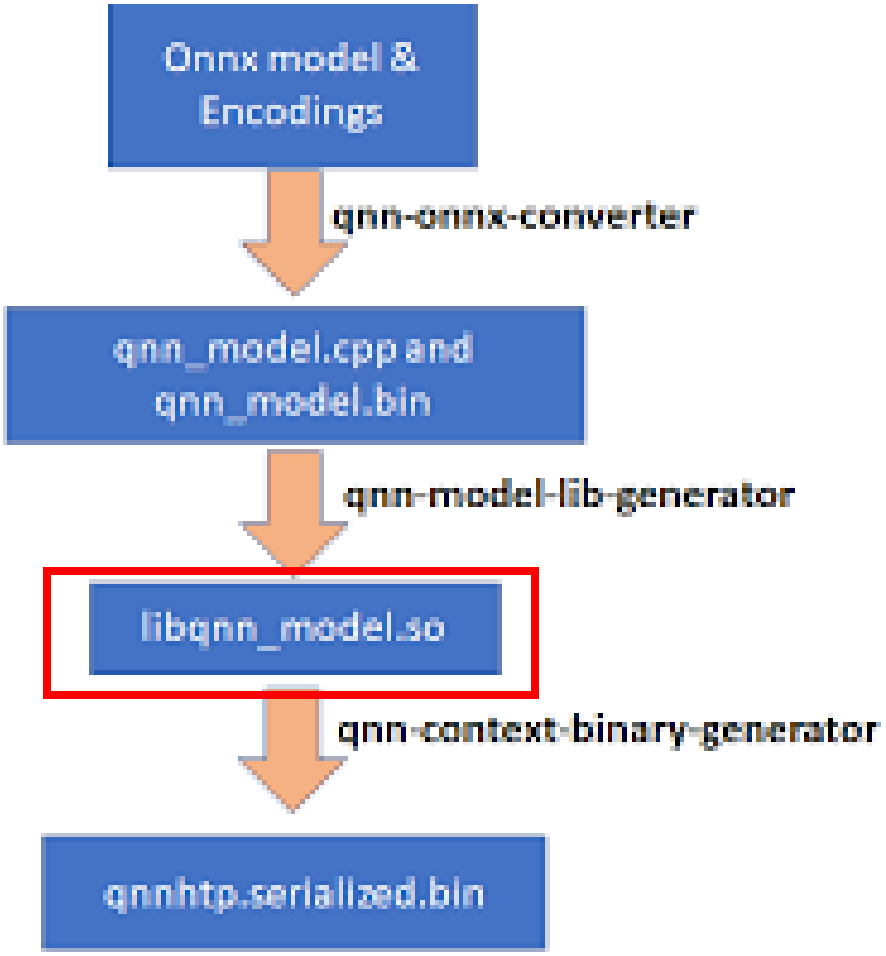
Well, No!

Then Are we doomed?

```
-rw-r--r--  1 1008 1000 4.6G Nov  6 10:28
model-00001-of-00002.safetensors
-rw-r--r--  1 1008 1000 313M Nov  6 10:26
model-00002-of-00002.safetensors
```

relocation R_X86_64_PC32 out of range: 2158227201 is not in [-2147483648, 2147483647]

This happens when running into architectural limitations. For example, in x86-64 PIC code, a reference to a static global variable is typically done with a R_X86_64_PC32 relocation, which is a 32-bit signed offset from the PC. That means if the global variable is laid out further than 2GB (2^31 bytes) from the instruction referencing it, we run into a relocation overflow.



QNN With ONNX

Model Partitioning

Model Compression

- 성능 손실
- 많은 시간필요

Model Quantization

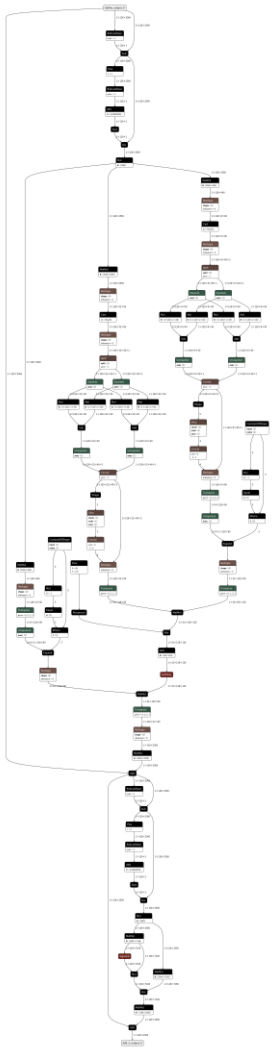
- 성능 손실
- 많은 시간 필요

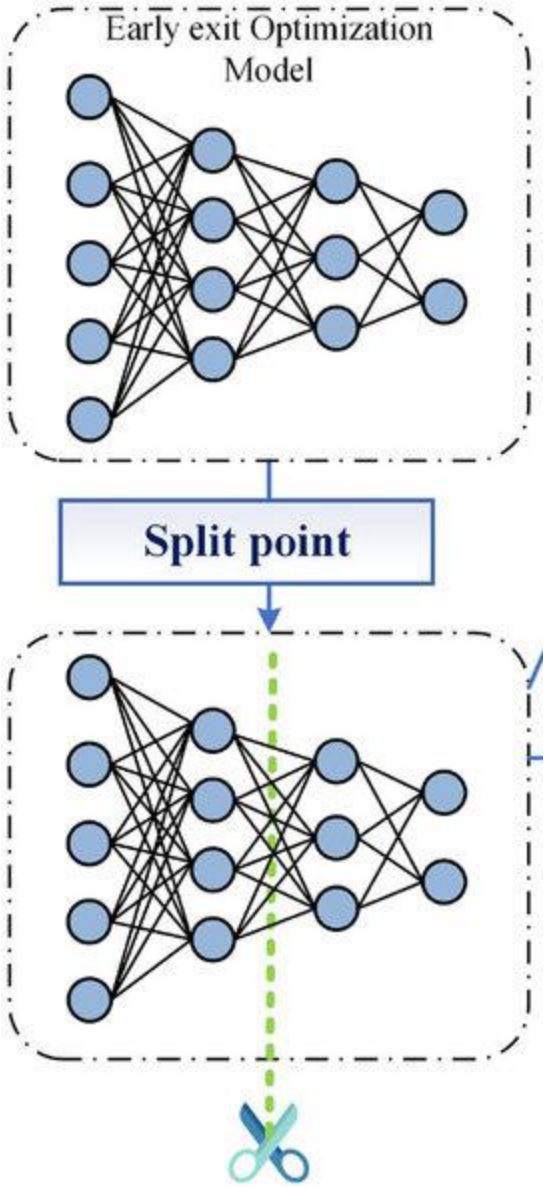
Hand-written QNN Code

- 많은 시간 필요
- Framework 작성임

Model Partitioning

- ONNX, torch 어디서든 가능
- 시간 절약





QNN With ONNX

Model Partitioning

named_children() [SOURCE]

Return an iterator over immediate children modules, yielding both the name of the module as well as the module itself.

```python
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self) -> None:
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

```python
modules = [child for child in a.children()]
embedding_output = modules[0].forward(torch.ones(1, max_token,
dtype=torch.long))

transformers = [mod for mod in modules[1]]
```

```python
import torch.nn as nn
import torch.nn.functional as F

class Model1(nn.Module):
    def __init__(self) -> None:
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)

    def forward(self, x):
        return F.relu(self.conv1(x))
```

```python
import torch.nn as nn
import torch.nn.functional as F

class Model2(nn.Module):
    def __init__(self) -> None:
        super().__init__()
        self.conv2 = nn.Conv2d(1, 20, 5)

    def forward(self, x):
        return F.relu(self.conv2(x))
```

QNN With ONNX

Model Partitioning

```python
def export_transformer(module, input, name: str):
  print("exporting...", name)
  exported = torch.export.export(module, (input, ))
  torch.onnx.export(exported, (input, ),\
                    name,\
                    verify=True, dynamo=True, optimize=True,\
                    external_data=False,\
                    do_constant_folding=True, opset_version=17)
  input = module(input)
```

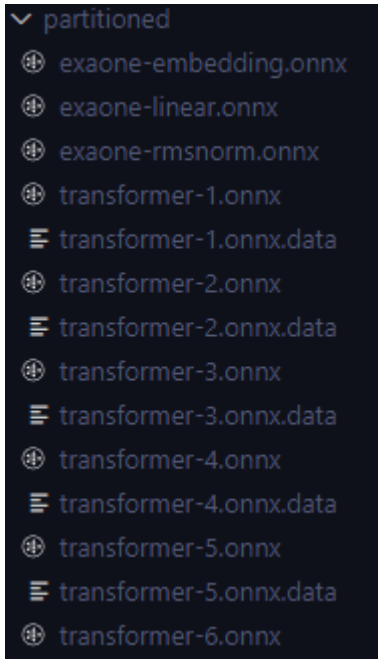
```python
class Layer(torch.nn.Module):
  def __init__(self, mod_list: List[TransformerSelfAttentionLayer]):
    super().__init__()
    for mod in mod_list:
      replace_rmsnorm_with_layernorm(mod)
    self.mod_list = torch.nn.Sequential(*mod_list)

  def forward(self, value):
    return self.mod_list(value)
```

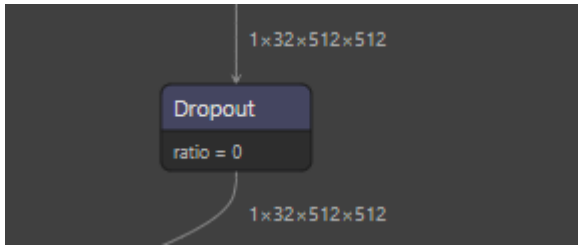
```python
export_transformer(Layer(transformers[:6]), embedding_output,
'partitioned/transformer-1.onnx')
export_transformer(Layer(transformers[6:12]), embedding_output,
'partitioned/transformer-2.onnx')
export_transformer(Layer(transformers[12:18]), embedding_output,
'partitioned/transformer-3.onnx')
export_transformer(Layer(transformers[18:24]), embedding_output,
'partitioned/transformer-4.onnx')
export_transformer(Layer(transformers[24:30]), embedding_output,
'partitioned/transformer-5.onnx')
export_transformer(Layer(transformers[30:]), embedding_output,
'partitioned/transformer-6.onnx')
```

QNN With ONNX

Node Replacing

Whole Model →

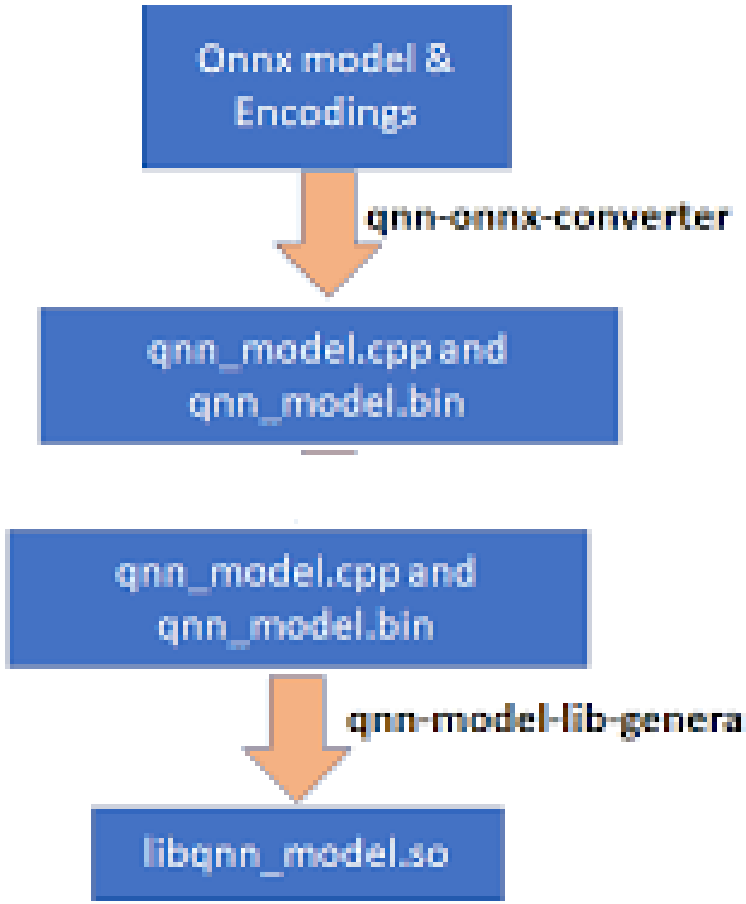





```python
import onnx
from onnx import helper, shape_inference
from onnx.tools import update_model_dims

from glob import glob

# Load the ONNX model
onnxs = glob('partitioned/*.onnx')


for file in onnxs:
    model = onnx.load(file)
    graph = model.graph

    # Identify nodes to remove and keep track of the connections
    nodes_to_remove = []
    for node in graph.node:
        if node.op_type == "Dropout":
            nodes_to_remove.append(node)

    for node in nodes_to_remove:
        input_tensor = node.input[0]  # Input tensor to Dropout layer
        output_tensor = node.output[0]  # Output tensor of Dropout layer

        # Find all nodes that use the Dropout's output as input
        for subsequent_node in graph.node:
            for i, input_name in enumerate(subsequent_node.input):
                if input_name == output_tensor:
                    # Redirect the input to the original input of the
Dropout
                    subsequent_node.input[i] = input_tensor

        # Remove the Dropout node from the graph
        graph.node.remove(node)

    # Save the modified model
    onnx.save_model(model, file + ".modified", all_tensors_to_one_file=True)
```

QNN With ONNX

Model Conversion




```bash
#/bin/bash

source /opt/qairt/2.28.0.241029/bin/envsetup.sh
#source /opt/qairt/2.26.0.240828/bin/envsetup.sh


rm -rf model-output/16a8w-layernorm

qnn-onnx-converter --preserve_io layout\
    -i ./partitioned/exaone-embedding.onnx.modified\
    -o model-output/16a8w-layernorm/embedding.cpp\
    --float_bitwidth 16 &
wait $(jobs -p)
```

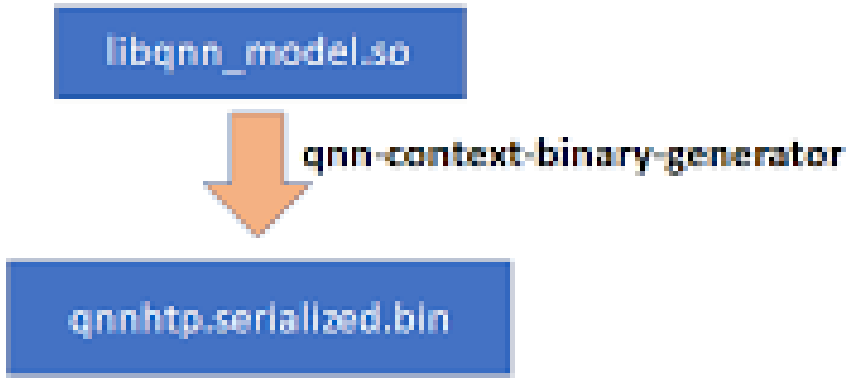
16a8w-layernorm
- embedding_net.json
- embedding.bin
- embedding.cpp
- linear_net.json
- linear.bin
- linear.cpp
- norm_net.json
- norm.bin
- norm.cpp

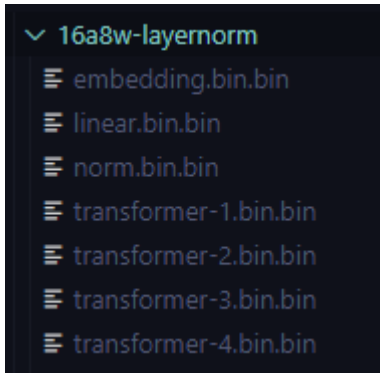
16a8w-layernorm / x86_64-linux-clang
- libembedding.so
- liblinear.so
- libnorm.so

```bash
source /opt/qairt/2.28.0.241029/bin/envsetup.sh

export ANDROID_NDK=/opt/android-sdk/ndk/26.3.11579264/
export PATH=$PATH:/opt/android-sdk/ndk/26.3.11579264/

rm -rf model-output/lib/16a8w-layernorm

qnn-model-lib-generator -c model-output/16a8w-layernorm/embedding.cpp\
    -b model-output/16a8w-layernorm/embedding.bin\
    -o model-output/lib/16a8w-layernorm/ \
    -t x86_64-linux-clang
```

QNN With ONNX

Model Conversion





- 모델 실행을 위한 바이너리는 준비 완료

```
source /opt/qairt/2.28.0.241029/bin/envsetup.sh

rm -rf ./model-output/bin/16a8w-layernorm

qnn-context-binary-generator \
    --config_file ./htp-backend-extension-data.json \
    --model ./model-output/lib/16a8w-layernorm/x86_64-linux-
clang/libembedding.so\
    --output_dir ./model-output \
    --binary_file ./bin/16a8w-layernorm/embedding.bin   \
    --backend /opt/qairt/2.28.0.241029/lib/x86_64-linux-clang/libQnnHtp.so
```

03
Tokenizer Code Review

Tokenizer

Tokenzier 종류

- Word-based Tokenizer
  - ✓ 단어 단위로 텍스트를 분리
  - ✓ Vocab size 매우 커짐
  - ✓ GPT-1 Tokenizer
- Subword-based Tokenizer
  - ✓ 단어를 더 작은 단위로 분리
  - ✓ GPT-2, BERT 등에서 사용
- Byte-based Tokenizer
  - ✓ 문자를 UTF-8 바이트로 변환
  - ✓ 고정된 어휘 크기
  - ✓ GPT-3 Byte-level BPE

최신 Tokenizer

- HuggingFace Tokenizer
  - ✓ Rust로 구현되어 Python보다 빠름
  - ✓ BPE, WordPiece, Unigram 등 지원
- SentencePiece
  - ✓ 언어 독립적, 텍스트를 사전 전처리 없이 처리 가능
  - ✓ 높은 압축 효율
- TikTokenizer
  - ✓ Byte-level BPE를 사용함
  - ✓ Rust로 구현되어 빠른 속도 보임

GPT2 Tokenizer

Tokenizer

Tokenizer?

Merge text : 단어의 우선 순위

Vocab json : 단어가 어느 값일지

: string to number

: number to string

Deterministic한 학습 결과에 의해 생성

Tokenizing part

Byte Encoder

Byte Decoder

Byte pair Encoding

BPE Ranks

```
void parse_merge_file(std::string const& merges_file);
void parse_vocab_file(std::string const& vocab_file);
```

```
// A Regular expression that handles text tokenization
static constexpr std::string_view pattern{
        "'s|'t|'re|'ve|'m|'ll|'d| ?\\p{L}+| ?\\p{N}+| ?[^\\s\\p{L}\\p{N}]+|\\s+(?!\\S)|\\s+"};
```

Mapping part

Encoder

Decoder

- 축약형 분리 (“ ‘s, ‘ll, ‘t” 등)
- 공백이 앞에 올 수 있는 단어 및 숫자 처리 (“[PAD]Hello” == “Hello”)
- 문자와 숫자를 개별 토큰으로 분리
- 특수 문자, 구두 점을 개별 토큰으로 분리
- N개의 공백도 별도의 토큰으로 분리

GPT2 Tokenizer

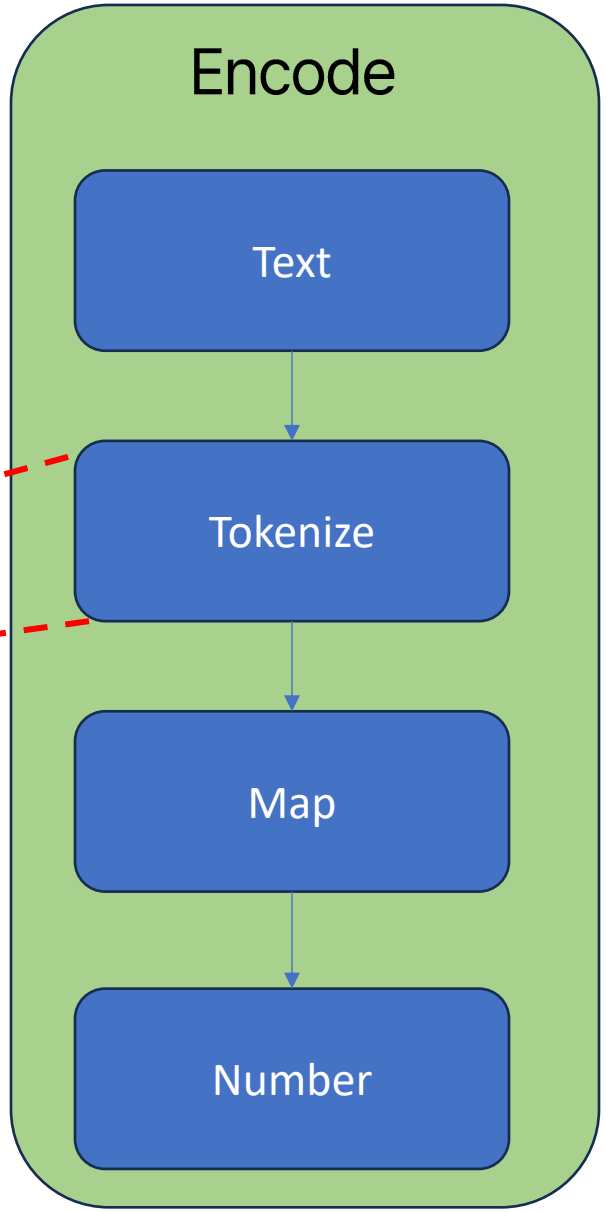
Tokenizer

Byte Encoder | Byte Decoder | Encoder | Decoder

Hello Hello World!

```
std::unordered_map<std::string, std::uint64_t> m_encoder;
std::unordered_map<std::uint64_t, std::string> m_decoder;

std::unordered_map<char, std::string> m_bytes_encoder;
std::unordered_map<std::string, char> m_bytes_decoder;
```

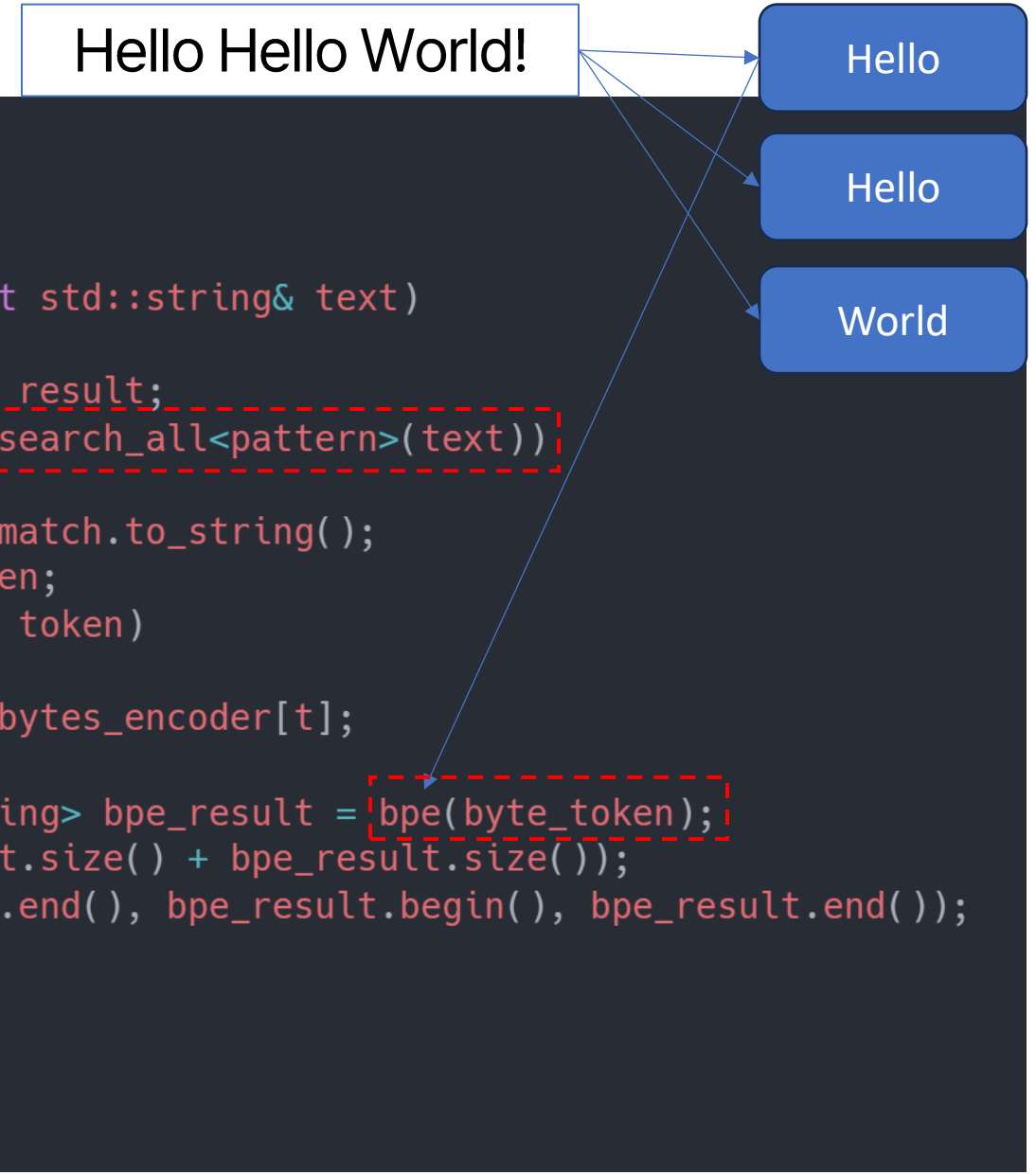
```
std::vector<int64_t>
GPT2Tokenizer::encode(const std::string& text)
{
    std::vector<std::string> tokens = tokenize(text);
    std::vector<int64_t>     token_ids;
    token_ids.reserve(tokens.size());
    std::transform(tokens.begin(),
                   tokens.end(),
                   std::back_inserter(token_ids),
                   [this](const std::string& token) { return m_encoder[token]; });
    return token_ids;
}
```

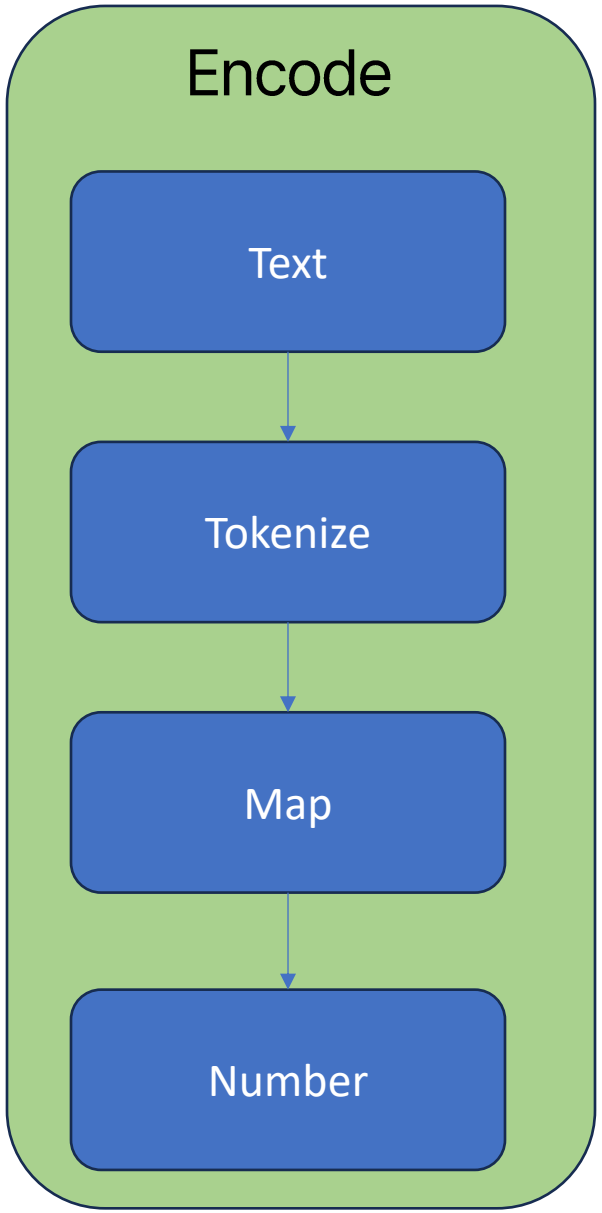


GPT2 Tokenizer

Tokenizer

```cpp
std::vector<std::string>
GPT2Tokenizer::tokenize(const std::string& text)
{
    std::vector<std::string> result;
    for (auto match : ctre::search_all<pattern>(text))
    {
        std::string token = match.to_string();
        std::string byte_token;
        for (const auto& t : token)
        {
            byte_token += m_bytes_encoder[t];
        }
        std::vector<std::string> bpe_result = bpe(byte_token);
        result.reserve(result.size() + bpe_result.size());
        result.insert(result.end(), bpe_result.begin(), bpe_result.end());
    }

    return result;
}
```





GPT2 Tokenizer

Byte Pair Encoding (Preprocess)

Hello

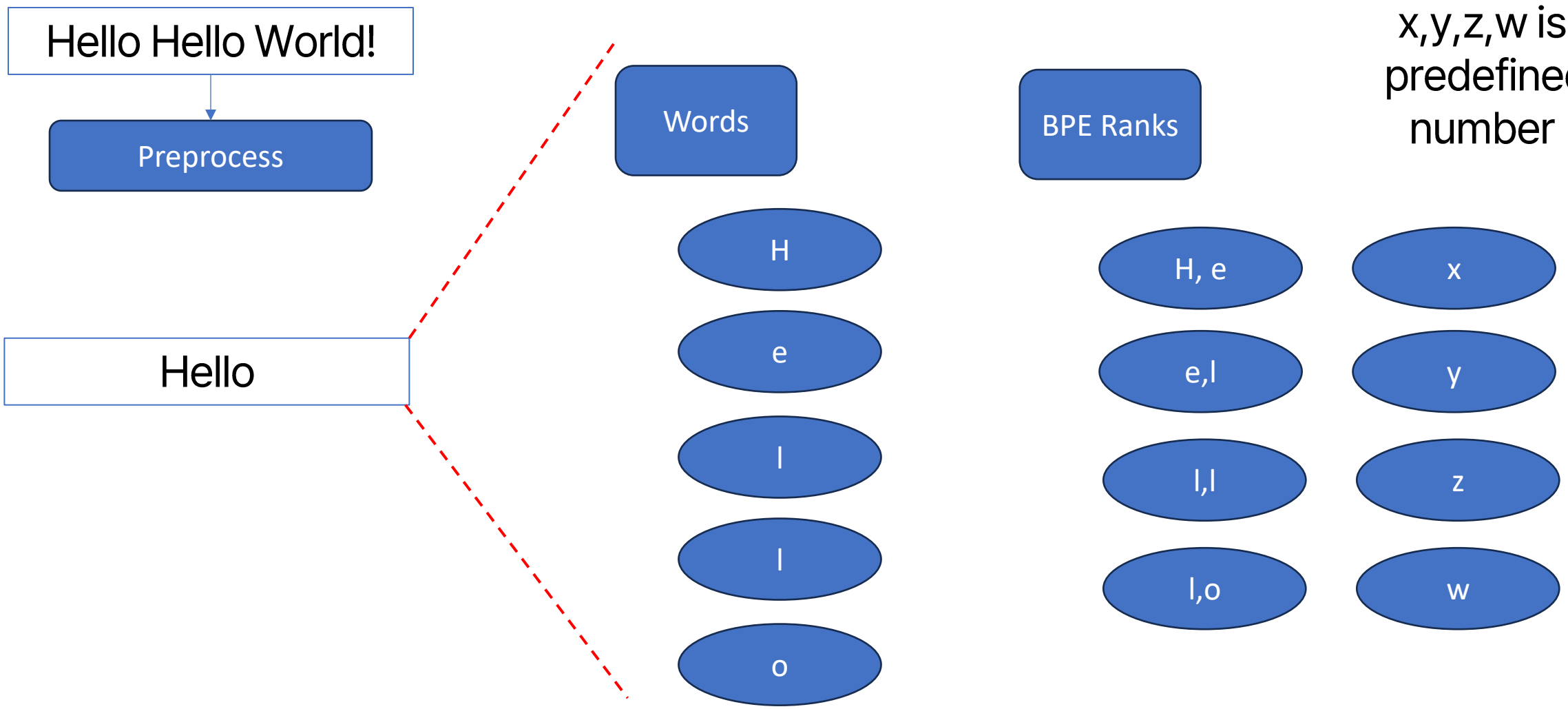
```cpp
std::vector<BPERanks::const_iterator> ranks;
// 벡터 word: 입력 문자열을 각 코드포인트(문자)로 나눈 결과를 저장
std::vector<std::string> word;
ranks.reserve(token.size() - 1); // ranks의 용량 예약: 최대 bigram 개수 (문자 수 - 1)
word.reserve(token.size());      // word의 용량 예약: 최대 문자 수

// 입력 문자열을 코드포인트 단위로 나누고 bigram 순위를 계산하는 부분
{
    size_t i = 0;
    while (true)
    {
        // 현재 코드포인트의 길이를 구함
        int length = codepoint_length(token[i]);
        // 다음 코드포인트의 길이를 구함
        int next_length = codepoint_length(token[i + length]);
        // 현재 문자와 다음 문자의 bigram 순위를 ranks에 추가
        ranks.push_back(
            m_bpe_ranks.find({token.substr(i, length), token.substr(i + length, next_length)}));
        // 현재 문자를 word에 추가
        word.push_back(token.substr(i, length));
        i += length;                    // 인덱스를 다음 문자로 이동
        if (i >= token.size()) break; // 문자열 끝에 도달하면 루프 종료
        if (i + next_length >= token.size())
        {
            // 마지막 문자를 word에 추가하고 종료
            word.emplace_back(token.substr(i, next_length));
            break;
        }
    }
}
```

BPE Ranks

```cpp
size_t
codepoint_length(const char c)
{
    if ((c & 0xf8) == 0xf0)
        return 4;
    else if ((c & 0xf0) == 0xe0)
        return 3;
    else if ((c & 0xe0) == 0xc0)
        return 2;
    else
        return 1;
}
```

GPT2 Tokenizer

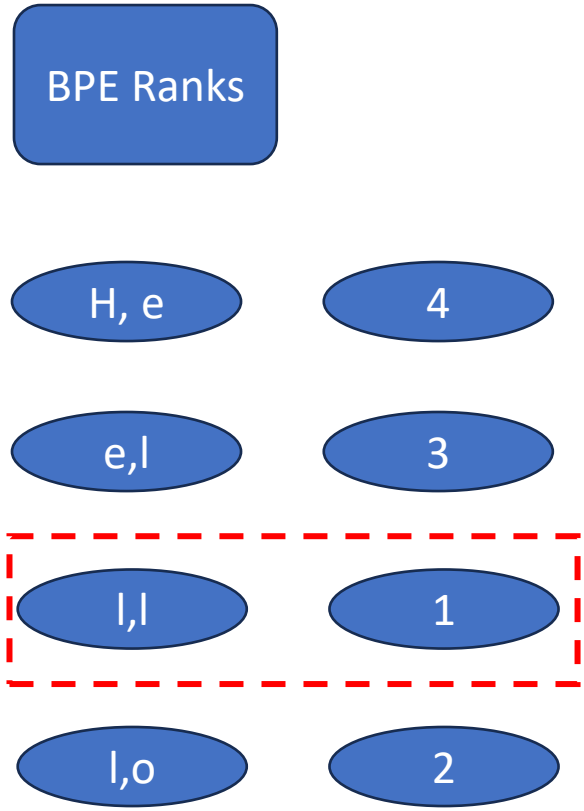
Byte Pair Encoding (Preprocess)



GPT2 Tokenizer

Byte Pair Encoding (Encoding)

```cpp
// ranks에서 가장 낮은 순위를 가진 bigram을 찾음
const auto bigram = std::min_element(
    ranks.begin(), ranks.end(), [this](const auto& lhs, const auto& rhs) -> bool {
        if (lhs == m_bpe_ranks.end() && rhs == m_bpe_ranks.end())
        {
            return false; // 둘 다 끝이면 false 반환
        }
        else if (lhs == m_bpe_ranks.end() || rhs == m_bpe_ranks.end())
        {
            return (lhs != m_bpe_ranks.end()); // 하나가 끝이면 유효한 다른 하나 선택
        }
        else
        {
            return lhs->second < rhs->second; // 둘 다 유효하면 순위가 낮은 것을 선택
        }
    });

if (bigram == ranks.end() || *bigram == m_bpe_ranks.end())
{
    // 더 이상 병합할 bigram이 없다면 종료
    break;
}
```



GPT2 Tokenizer

Byte Pair Encoding (Encoding)

H,e,l,l,o

Hello

For 1st iteration

l,l

H, e

l,l -> ll

```cpp
const auto [first, second] = (*bigram)->first; // 병합할 bigram의 첫 번째와 두 번째 문자
std::vector<std::string> new_word;             // 새로운 단어 벡터 생성

size_t i = 0;
while (i < word.size())
{
    // 첫 번째 문자를 찾음
    const auto wordIterator = std::find(word.begin() + i, word.end(), first);
    if (wordIterator == word.end())
    {
        // 첫 번째 문자가 없으면 나머지 문자를 모두 new_word에 추가하고 종료
        std::copy(word.begin() + i, word.end(), std::back_inserter(new_word));
        break;
    }

    // 첫 번째 문자 이전의 모든 문자를 new_word에 추가
    std::copy(word.begin() + i, wordIterator, std::back_inserter(new_word));
    i = std::distance(word.begin(), wordIterator); // 현재 인덱스를 업데이트

    if (word[i] == first and i < word.size() - 1 and word[i + 1] == second)
    {
        // 첫 번째 문자 다음에 두 번째 문자가 있으면 병합하여 new_word에 추가
        new_word.push_back(first + second);
        i += 2; // 두 문자를 병합했으므로 인덱스를 두 칸 이동
    }
    else
    {
        // 병합하지 않는 경우 현재 문자를 new_word에 추가
        new_word.push_back(word[i]);
        i += 1; // 인덱스를 하나 증가
    }
}
word = std::move(new_word); // word를 new_word로 업데이트
```

GPT2 Tokenizer

Tokenizer

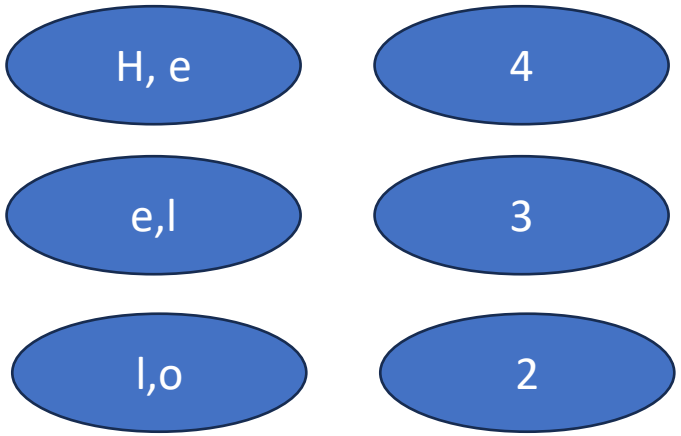
Remove merged bigram

```
if (word.size() == 1)
    break; // word의 크기가 1이면 더 이상 병합할 수 없으므로 종료
else
{
    // 새로운 bigram 순위를 ranks에 업데이트
    for (size_t i = 0; i < word.size() - 1; ++i)
    {
        ranks[i] = m_bpe_ranks.find({word[i], word[i + 1]});
    }
    ranks.resize(word.size() - 1); // ranks의 크기를 업데이트
}
```

What if word is “Hallo”?

BPE Ranks

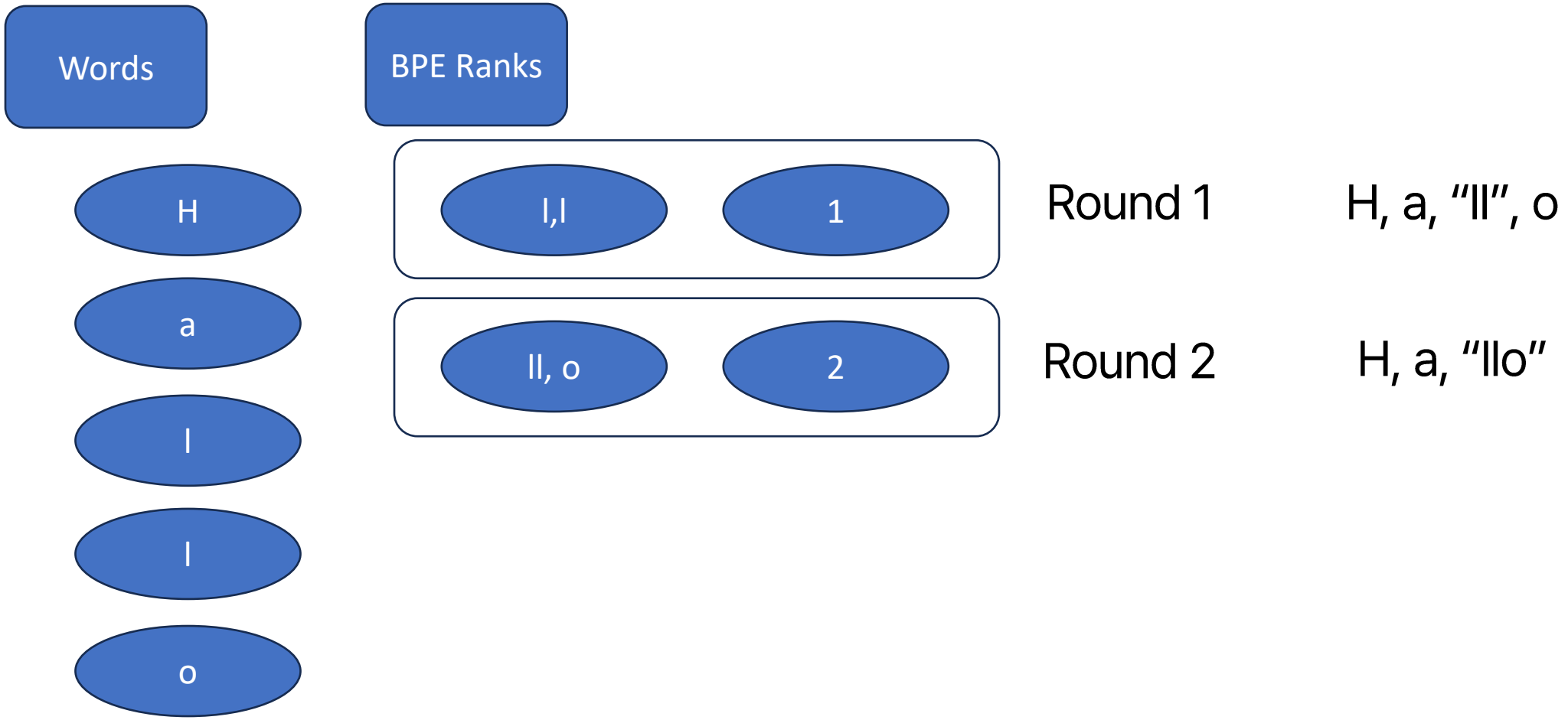
x,y,z,w is predefined number



GPT2 Tokenizer

Tokenizer

What if word is “Hallo”



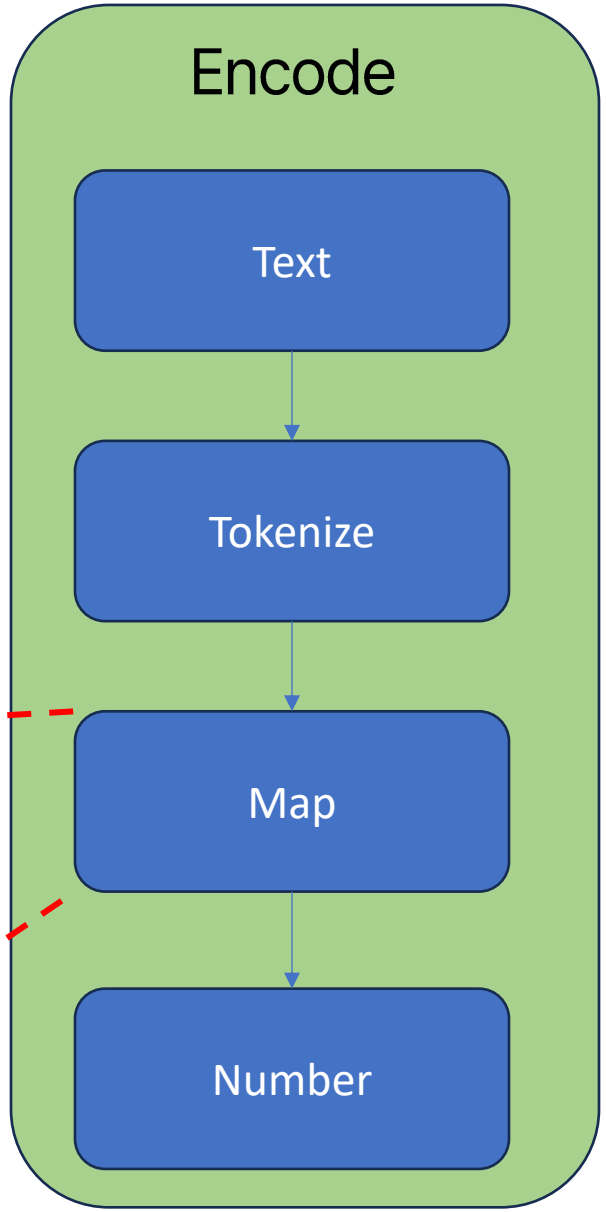
GPT2 Tokenizer

Tokenizer

Byte Encoder | Byte Decoder | Encoder | Decoder

```
std::unordered_map<std::string, std::uint64_t> m_encoder;
std::unordered_map<std::uint64_t, std::string> m_decoder;

std::unordered_map<char, std::string> m_bytes_encoder;
std::unordered_map<std::string, char> m_bytes_decoder;
```

```
std::vector<int64_t>
GPT2Tokenizer::encode(const std::string& text)
{
    std::vector<std::string> tokens = tokenize(text);
    std::vector<int64_t>     token_ids;
    token_ids.reserve(tokens.size());
    std::transform(tokens.begin(),
                   tokens.end(),
                   std::back_inserter(token_ids),
                   [this](const std::string& token) { return m_encoder[token]; });
    return token_ids;
}
```



End.